Безопасность файлов и метаданных в СУБД Firebird

Geoff Worboys

25 декабря 2005 - Версия документа 0.5.ru

Перевод документа на русский язык: Павел Меньщиков

Содержание

Введение

Если Вы находитесь в замешательстве, пребывая на этой странице, и/или хотите узнать подробнее о СУБД Firebird, обратитесь к www.firebirdsql.org

В этой статье описываются вопросы, связанные с безопасностью файлов базы данных СУБД Firebird, в особенности безопасности метаданных, хранящихся в этих файлах. Статья была написана в ответ на большое количество вопросов, касающихся указанной темы, в листах поддержки СУБД Firebird. Статья избегает технических подробностей темы. Для выяснения способов обеспечения безопасности файлов определенной операционной системы обратитесь к соответствующей документации по используемой операционной системе; для выяснения способов обеспечения безопасности объектов базы данных СУБД Firebird обратитесь к документации на сайте, ссылка на который дана чуть выше, или приобретите книгу Хелен Борри (Helen Borrie) *The Firebird Book* (в скором будущем планируется выпуск переведенной на русский язык версии этой книги, которая будет называться *Firebird: Руководство разработчика баз данных* и выйдет в издательстве «БХВ-Петербург» - прим. перев.).

Исходные условия

Чтобы получить доступ к базе данных СУБД Firebird, приложению (пользователю) необходимо подключиться к процессу сервера Firebird. При получении запроса на соединение процесс сервера удостоверяет пользователя по базе данных безопасности (security database). После успешного удостоверения сервер разрешает приложению (пользователю) произвести доступ к любой базе данных, а затем сервер использует роли и права пользователей, определенные в самой базе данных, чтобы обеспечить тонкую настройку системы доступа к объектам базы данных.

Никогда при подключении к базе данных пользователю нет необходимости иметь прямой доступ к, собственно, файлу базы данных. Все соединения и весь доступ происходит через процесс сервера баз данных, который выполняет обращение к файлу базы данных по необходимости, с целью выполнить запросы пользователя. Сервер является тем «устройством», которое ограничивает или разрешает доступ зарегистрированным пользователям в соответствии с правами этих пользователей, определенными для каждого конкретного пользователя в каждой конкретной базе данных.

> **Замечание**
>
> «Встроенная» версия СУБД Firebird имеет другой механизм работы системы безопасности, и она не пригодна для безопасных инсталляций (установок и настроек) СУБД и базы данных. Факт существования «встроенной» версии сервера практически не изменяет вопросы безопасности, обсуждаемые в этой статье, а только лишь выделяет важность использования эффективных методов обеспечения безопасности всего рабочего окружения в безопасной инсталляции СУБД и баз данных. «Встроенная» версия СУБД обсуждается более подробно далее.

Каждая установленная СУБД Firebird имеет предопределенного пользователя с именем SYSDBA. Этот пользователь является администратором, и имеет неограниченный доступ ко всем базам данных, доступным СУБД. Права, определенные в конкретной базе данных, игнорируются, если подключение к базе данных осуществлется пользователем SYSDBA. Когда Вы впервые копируете базу данных на сервер, Вы можете использовать учетную запись SYSDBA для настройки привилегий в соответствии с определенными для СУБД пользователями и требованиями сервера. Однако это также означает, что если у кого-то есть прямой доступ к файлу базы данных, то он без труда сможет скопировать этот файл с сервера, где пароль пользователя SYSDBA неизвестен, на другой сервер, где пароль будет известен, что приведет к возможности неограниченного доступа к содержимому базы данных.

Как Вы видите, безопасность СУБД Firebird исходит из предположения о том, что серверный процесс будет работать в достаточно безопасном окружении. Сама СУБД Firebird не предпринимает никаких мер для обеспечения внешней безопасности. Как только некто получает физический доступ к файлу базы данных, не существует способов предотвратить доступ ко всем данным (в том числе, метаданным) этого файла.

> **Замечание**
>
> «Достаточность» безопасности определяется уровнем безопасности, необходимым и/или требуемым конкретной инсталляции (установке) СУБД. Этот уровень весьма сильно изменяется от инсталляции к инсталляции.

Это означает, что для обеспечения разумной безопасности, процесс инсталляции должен удостовериться, что файл базы данных достаточно защищен. В большинстве случаев, это равносильно тому, что процесс сервера Firebird должен работать с правами отдельного пользователя (учетной записи) операционной системы, и только эта учетная запись (и еще, быть может, администратор операционной системы) должна иметь прямой доступ на уровне операционной системы к файлам баз данных. Файлы базы данных никогда не следует располагать в папках, к которым открыт доступ из сети или доступ любых локальных пользователей, кроме специально оговоренного персонала.

Чтобы меры безопасности были эффективны, также очень желательно, чтобы серверное оборудование было установлено в месте, куда ограничен физический доступ посторонних лиц, дабы не допустить несанкционированный доступ к жестким дискам сервера.

Однако приведенное объяснение вовсе не обязательно будет полезно разработчикам, которые создают базы данных для распространения и установки на удаленных серверах, но хотели бы защитить свою интеллектуальную собственность, реализованную и находящуюся в самих базах данных. Такие случаи могут включать метаданные (структуры таблиц и отношений, исходные и скомпилированные тексты хранимых процедур и триггеров), а также некоторые данные, содержащиеся в таблицах. Эти случаи и являются основным фокусом этой статьи.

Проблема

Разработчик создает базу данных (и, обычно, сопутствующее клиентское программное обеспе-

чение) для установки на удаленных серверах и клиентах. Обычно серверы обслуживаются людьми, некоторые из них имеют полный доступ к системе, где работает СУБД Firebird и где находятся базы данных, - эти люди осуществляют резервное копирование баз данных и другие процедуры обслуживания. Как описано в предыдущем разделе, прямой доступ к файлу базы данных обеспечивает возможность полного и неограниченного доступа к информации, содержащейся в базе данных, - метаданным и данным.

В таких случаях разрабочик может не иметь доверия к пользователям удаленных инсталляций: опасения могут быть в том, что пользователи способны воспользоваться методом обратного инжиниринга базы данных, а затем воспользоваться результатами в своих целях; либо опасения могут быть в том, что среда работы сервера (оборудования и программ) не будет обеспечивать нужный уровень безопасности, что может привести к неавторизованому доступу к базе данных.

Все это ведет к общим вопросам в листах поддержки СУБД Firbird:

«Я хочу...»

> «...защитить дизайн моей базы данных (структуры таблиц, хранимые процедуры, триггеры и т.д.) от доступа всеми пользователями базы данных на удаленных, не контролируемых мной, инсталляциях. Как я могу достичь этого с СУБД Firebird?»

«Я хочу...»

> «...предотвратить доступ пользователей удаленных, не контролируемых мной, инсталляций к данным определенных таблиц. Эти таблицы содержат проприетарную информацию, которая используется исключительно для внутреннего использования приложением.»

СУБД Firebird (по крайней мере, до версии 1.5) не содержит встроенных механизмов шифрования. Простым ответом на оба вопроса является такой, что с имеющимися средствами СУБД реализация пожеланий невозможна. Пользователь, получивший доступ к файлу базы данных, получает доступ ко всем данным этого файла.

В первом случае, не существует «обходных путей», так как серверный процесс должен иметь доступ к файлу базы данных и метаданным. Во втором случае, возможно на клиентской стороне реализовать механизм шифрования/дешифрования данных, но в таком случае у Вас пропадает возможность добавлять эффективные индексы и использовать возможности поиска - управление остается главной проблемой (более подробно далее).

Решение

Существует единственный способ решить указанные ранее вопросы: размещайте базы данных и СУБД на Ваших собственных серверах, и предоставляйте к ним доступ клиентам, например, через интернет и т.п. Возможности терминальных серверов (terminal servers) (Windows или Linux/Unix) так же можно использовать для реализации таких требований.

Таким образом, у Вас есть контроль над базой данных, и Вы можете ограничивать доступ к различным функциям и структурам баз данных, используя имеющиеся в СУБД Firebird методы

обеспечения безопасности (роли, привилегии и т.д.).

Трудности

Стоит отметить, что и при таком подходе существуют некоторые трудности, если Ваша цель - защита структуры базы данных.

Необходимость в уровнях доступа

Различные библиотеки доступа к базе данных запрашивают метаданные, такие как первичные ключи, домены (domain) и т.п., чтобы облегчить процесс разработки клиентских приложений. Поэтому Вы можете открыть для себя факт, что Вы не можете предотвратить доступ пользователей к метаданным без прекращения предоставления приложению той информации, которая ему необходима.

Это означает, что Вам необходимо выбирать между разрешением открывать пользователям детали метаданных и пользоваться возможностями развитого интерфейса доступа к данным, и проведением значительного времени за разработкой приложений, которые используют менее совершенные библиотеки доступа к данным.

«Утечки» подразумеваемые и явные

Существует вопрос клиентских приложений, которые, по сути, организуют «утечку» информации о структуре базы данных, с которой эти приложения работают. Очень редко можно встретить в приложениях, где главную роль играет база данных, интерфейс, не сильно отражающий детали структур, на основе которых он построен.

Некоторые детали можно скрыть использованием представлений (views) и хранимых процедур-выборок (selectable stored procedures), но создание такого функционала, чтобы скрыть структурную информацию, является упражнением в разочарованиях. Вероятнее всего, это бесполезное занятие, так как часть деталей строения базы данных все равно станет доступной публике, как бы Вы ни старались.

Защита пользовательских данных

Перед тем, как продолжить обсуждение вопросов шифрования данных в СУБД Firebird, хочется обратить Ваше внимание на то, что пользователи могут защищать свои данные при помощи шифрования. Это не поможет разработчикам защитить свою информацию от легальных пользователей базы данных, но поможет удовлетворить требование заказчиков, желающих увеличить безопасность баз данных.

В некоторых (офисных) случаях бывает непрактичным размещать сервер с СУБД Firebird в реально безопасном окружении: в течение рабочего дня очень маловероятно, что кто-то сможет

проникнуть в офис и получить доступ к компьютеру с целью копирования файлов базы данных (или украсть компьютер или жесткий диск, чтобы получить доступ к файлам позже). Однако в остальное время (ночью и/или в выходные) безопасность сервера может быть под вопросом, и это должно рассматриваться в конкретных случаях.

Шифрование

В то время, как СУБД Firebird не предоставляет встроенных средств шифрования данных, существует несколько замечательных продуктов, чтобы это делать. Вы можете установить программное обеспечение, которое создает защищенные тома (volumes) на Вашем компьютере, и Вы размещаете базы данных (и другие конфиденциальные данные) на этих томах. Когда компьютер выключен, все данные таких томов существуют в зашифрованном виде, и Вы не сможете получить к ним доступ без ключа. При загрузке компьютера Вам необходимо смонтировать (mount) защищенные тома и указать ключ для дешифрования, чтобы получить доступ к данным. Этот дополнительный, и, обязательно, собственноручный, шаг при запуске компьютера может быть неудобным, но он может обеспечить превосходную безопасность для «не сильно обслуживаемых» компьютерных систем.

Программным обеспечением с указанными функциями являются TrueCrypt (www.truecrypt.org), Bestcrypt от Jetico (www.jetico.com) и PGPDisk (www.pgpi.org/products/pgpdisk/ (заметьте, что эта ссылка ведет на старую бесплатную версию программы, а сайт содержит ссылки и на более новые платные версии продукта). Существуют и другие программные продукты... Последними двумя пользуюсь я сам.

Почему СУБД Firebird не обеспечивает шифрование?

Из-за вышеприведенных причин обычным среди пользователей является пожелание для будущих версий СУБД Firebird добавить возможность шифрования метаданных, результатов запросов пользователей к базе данных и даже базы данных целиком. Не будучи основным разработчиком СУБД, я не могу сказать категорично, что этого не случится. Однако суть вопроса заключается не в том, является шифрование практичным и/или полезным, а в том, обеспечат ли решение указанных проблем методы управления ключами шифрования.

Шифрование является хорошим настолько, насколько хорош закрытый ключ (secret key), используемый для дешифрования. Оно может быть хуже, но не лучше. Существует несколько замечательных алгоритмов шифрования, которые можно использовать. Когда используется хорошие алгоритмы шифрования, скорее всего, атаки будут предприниматься на поиски методов доступа к ключу, а не на само шифрование.

Как могло бы работать шифрование?

Давайте представим, как бы обстояли дела, если бы СУБД Firebird шифровала метаданные базы данных...

Перед тем, как СУБД сможет подключиться к базе данных, ей необходимо предоставить ключ для дешифрования данных. Предоставление ключа дешифрования пользователю было бы бес-

смысленным, так как это действие вернуло бы нас обратно к исходной проблеме. Тогда, скорее всего, при любом перезапуске сервера пользователям нужно звонить разработчику, чтобы он некоторым образом указал серверу ключ. Даже если бы это было практически реализуемо, такой подход вовсе не обязательно решил бы проблему. Например, заказчики могут устанавливать на свои серверы программы слежения, которые могут зафиксировать все нажатия клавиш.

Существуют решения предоставления ключа процессу дешифрования на основе некоторого оборудования. Но, опять же, это оборудование будет находиться у Ваших заказчиков/клиентов, и Вы не сможете предотвратить использование этого оборудования клиентом, чтобы получить доступ к базе даных с другого сервера баз данных, где известен пароль пользователя `SYSDBA`.

СУБД Firebird является проектом с открытым исходным кодом. Если бы возможности шифрования были встроены, или использовались бы дополнительные библиотеки шифрования с открытым исходным кодом (plug-ins), то для пользователя было бы вполне возможно создать свою версию СУБД или библиотеки, которая не только выполняла бы необходимые шифрование и дешифрование для доступа к базе данных, но и, например, сохраняла бы ключ в файл, или напрямую сохраняла незащищенные данные. Разработчик, не обладая контролем над сервером, не в состоянии ни определить, ни предотвратить такие действия.

Вы можете рассматривать вариант создания собственной версии СУБД Firebird, в которой ключ дешифрования будет скрыт в исполняемом файле. Однако существуют декомпиляторы. Не составит большого труда обнаружить ключ, просто сравнив декомпилированные версии Вашего исполнимого файла СУБД и обычной версии.

Существуют продукты для работы с базами данных, чья цель - обеспечение сильного шифрования. Возможно, шифрование и является сильным, но, если отсутствует хорошая система управления ключами (key management), то шифрование не достигнет желаемого эффекта. Это может лелеять в Вас веру в то, что Вы защищены, но Вам необходимо поближе познакомиться с системой управления ключами, чтобы выяснить, так ли это на самом деле.

Горькой правдой является то, что даже при однократной потере контроля над оборудованием, где производится шифрование и дешифрование, положение становится неясным. Если ключ дешифрования не может быть в полной безопасности, то даже хорошее шифрование становится немного больше, чем «безопасностью по неясности» (security by obscurity).

Ограничение распространения данных

Некоторые просят шифровать данные базы данных, и, таким образом, хотят попытаться ограничить распространение данных. Их устраивает, если авторизованный пользователь имеет доступ к данным для их просмотра, но они бы хотели ограничить возможности таких пользователей передавать данные другим.

Представьте, что все ранее описанные проблемы с управлением ключами решены, так что пользователю нет смысла делать обычную копию базы данных. В этом случае пользователь может создать простую программу, которая будет извлекать интересующие его данные (используя легально установленный сервер) и записывать их в свой собственный файл или базу данных.

Думаю, что в будущем СУБД Firebird сможет обеспечивать некоторую систему опознавания (аутентификации) приложений, что позволит ограничить указанную форму извлечения данных, однако бОльшая часть проблем остается. Если Вы не имеете контроля над сервером, то не сможете предотвратить установку версии сервера, не требующую аутентификации.

Исключение доступа SYSDBA

Периодически люди предлагают исключить возможность доступа пользователя `SYSDBA` к объектам базы данных. Идея заключается в том, что при переносе базы данных на другой сервер, где пароль пользователя `SYSDBA` известен, знание пароля не поможет получить доступ к данным, потому что у пользователя `SYSDBA` к базе данных доступ будет запрещен. Некоторые сообщают об небольших успехах: создается роль SQL с именем `SYSDBA`, и эта роль не имеет прав для доступа к объектам базы данных.

Однако на самом деле это не решает проблему. Файл базы данных может быть исследован при помощи утилит просмотра двоичных файлов (hex viewer) или подобных утилит, что даст список доступных имен пользователей. Особенно полезной окажется информация о владельцах объектов базы данных. Как только станет известна эта информация, пользователи с найденными именами могут быть добавлены в базу данных пользователей нового сервера и использоваться напрямую.

Можно поступить еще проще, если использовать встроенную версию (embedded version) СУБД Firebird (см. далее), или скомпилировать Вашу собственную версию сервера Firebird, которая будет игнорировать органичения системы безопасности SQL.

Собственное имя для SYSDBA

Были предложения позволить изменять имя администратора `SYSDBA` на другое. Это может немного помочь противостоять прямым сетевым атакам для выяснения пароля администратора, так как при таких атаках придется подбирать не только пароль, но еще и имя, но это не поможет защитить систему от злоумышленника, имеющего прямой доступ к файлу базы данных.

Удаление исходного кода хранимых процедур и триггеров

При создании хранимой процедуры или триггера СУБД Firebird сохраняет практически полную копию исходного кода хранимой процедуры или триггера вместе с «откомпилированной» копией, называемой также BLR (Binary Language Representation). Откомпилированная копия выполняется сервером, исходный код им не используется.

Некоторые разработчики пытаются защитить по крайней мере некоторые из метаданных, удаляя исходный код из базы данных перед тем, как передавать базу данных заказчикам (простое прямое обновление соответствующих полей системных таблиц). Я не рекомендую удалять исходные коды по двум причинам...

1. BLR является довольно несложным преобразованием исходного кода. Не составит особого труда декодировать BLR обратно в понятное представление. Такое декодирование не будет содержать комментариев и форматирования, но обычно используемые операторы SQL не являются слишком сложными, так что это, скорее всего, не будет проблемой. Соответственно, защита, предоставляемая удалением исходного кода, не является значимой.

2. Исходный код может пригодиться для других целей. Можно сразу вносить поправки в базу данных: нет необходимости брать исходный код где-то еще, а потом не забыть снова удалить его. Исходный код также используется различными утилитами, такими как моя собственная программа DBak - альтернатива стандартной программе резервирования "gbak". Я пока не пытался создавать декодер для BLR, поэтому DBak рассчитывает на доступность исходных кодов, чтобы строить скрипты DDL (data definition language, язык определения данных), которые реконструируют базу данных.

Встроенный сервер Firebird

Существует специальная версия СУБД Firebird, называемая «встроенной». Это клиентская библиотека, которая содержит, собственно, сам сервер. Когда приложение обращается к такой библиотеке, оно загружает сервер и разрешает прямой доступ к любой базе данных, имеющейся на локальном компьютере. Эта версия сервера не использует базу данных безопасности, содержащую информацию о пользователях сервера. Имя пользователя, указанное при «входе» (проверка пароля при этом не производится), используется для управления доступом пользователя к объектам базы данных (с помощью механизма разрешений/запрещений SQL). Если именем пользователя является `SYSDBA` или имя владельца базы данных, возможен неограниченный доступ.

Функциональность встроенной версии удобна для разработчиков, которые хотят создать легко распространяемые приложения для одного пользователя, не требующие использования условий безопасности.

Из короткого описания может показаться, что наличие установленной встроенной версии сервера на компьютере, где установлена полная версия сервера, обслуживающего другие базы данных, само по себе является большой проблемой с безопасностью. На самом деле, риск нарушения безопасности не больше, чем когда встроенный сервер не был бы установлен вовсе.

Когда приложение загружает встроенный сервер, то сервер работает в контексте безопасности приложения (и, соответственно, пользователя ОС). Это означает, что встроенный сервер сможет получить доступ только к тем файлам баз данных, к которым пользователь операционной системы имеет прямой доступ. Разрешение обычному пользователю устанавливать программы на сервере, который должен быть обезопасен, является плохой идеей в любом случае. Однако, если Вы указали необходимые права доступа к файлам баз данных сервера, то встроенный сервер доступа к ним не получит.

Опасность исходит от других программ, которые пользователь может установить.

Сам факт существования встроенного сервера только лишь подчеркивает, что возможно сде-

лать, если получить прямой доступ к файлу базы данных, особенно для проектов с открытым кодом. Если бы такой версии сервера еще не было, то, несомненно, кто-то смог бы сделать версию полного сервера, подобную встроенной версии.

Другие виды «безопасности по неясности»

Предлагаются и различные другие формы «безопасности по неясности» (security by obscurity). Например, специальные события, возникающие в моменты входа/подключения и отключения пользователя, и вызывающие пользовательские функции для предотвращения или запрещения доступа. Такие функции могут ограниченно использоваться в системах с закрытым исходным кодом, где неясность реализации помогает скрыть механизмы защиты информации. Для проекта с открытым исходным кодом обходным путем может стать компиляция Вашей собственной версии сервера, которая не будет реагировать на события или не будет содержать код, предотвращающий доступ. Сложно внести неясность в проект с открытым исходным кодом.

Подумайте также, что происходит, когда Вы распространяете исполняемые файлы Вашей программы. Скомпилированные программы являются замечательным примером неясности (точнее, временной невыясненности). Шифрование обычно не используется, все алгоритмы и данные находятся там, чтобы быть проанализированными кем-то, кто имеет достаточно времени и знаний, и, конечно же, на помощь ему придут декомпиляторы. Когда будет известно, с какими библиотеками скомпилирована Ваша программа, процесс выделения Вашего «секретного» кода пойдет намного быстрее. Вы писали в Borland, Microsoft или кому-то еще с просьбой, чтобы они каким-то образом шифровали скомпилированные исполняемые файлы?

Приемлемая низкая безопасность

Пока мои комментарии относились к идее сильной безопасности, и я предполагаю, что про концепцию «безопасности по неясности» (security by obscurity) было написано с некоторым презрением. Но иногда слабая безопасность - все, что Вам необходимо. Иногда данные могут не быть столь ценными. Вам нужно не допустить случайных попыток обзора данных, и, по крайней мере, затруднить процесс доступа к данным для опытного вора.

Я сам использовал такие схемы в нескольких местах. Часто нет смысла использовать Twofish, AES или что-то еще в таких схемах, так как все это относится к сильному шифрованию. Эти алгоритмы добавляют приличный объем вычислений и усложенений, относящихся к сохранению сильной безопасности. Простой XOR (исключающее ИЛИ) с использованием некоторых известных строк (ключей) может быть достаточным. Если ключ каким-то образом можно узнать, то не имеет значения, использовали Вы слабое или сильное шифрование - игра все равно будет окончена.

Замечание

Наиболее простые алгоритмы XOR можно сломать без особого труда. Обратитесь к хорошему справочнику по шифрованию для получения дополнительной информации на этот счет.

Выбор «неясности»

Основной вещью в «безопасности по неясности» является неясность! Если бы в СУБД Firebird был реализован некоторый вид шифрования при чтении с диска и записи на диск, то это не было бы неясностью, потому что Firebird является проектом с открытым исходным кодом. Почти не заняло бы времени выяснение ключа шифрования, и вся защита была бы вскрыта.

Поэтому, если бы Вам понадобилась такая функция, то Вам нужно было бы взять исходные коды Firebird, добавить Ваш код, вносящий некоторую неясность в методы чтения с диска и записи на диск, и скомпилировать Ваш собственный вариант сервера Firebird. (Такой код может быть декомпилирован, но для этого нужен довольно серьезный вор.)

Перед тем, как Вы это сделаете, осознайте, решит ли это на самом деле Вашу проблему: если пользователь вместе с базой данных скопирует и Вашу версию СУБД, или если пользователь по-прежнему сможет извлечь секреты прямо при работе с Вашей версией сервера.

Философский аргумент

Существует философский вопрос, почему Вы хотите выбрать СУБД с открытым кодом для создания базы данных с закрытым кодом? Многие содействуют проекту своей твердой верой в то, что открытый исходный код является наилучшим способом работы с программным обеспечением.

Однако, если касаться вопроса хранения данных разных пользователей, то здесь я являюсь сильным приверженцем той точки зрения, что пользователи должны настаивать на возможности доступа к своим собственным данным, что зачастую включает в себя необходимость понимания структур и процессов, которые Вы разработали (метаданные). Если Вы покинете бизнес или каким-то образом до Вас нельзя будет «добраться», для пользователя, возможно, будет критически важно хотя бы извлечь свои собственные данные (в нужном им формате), чтобы перейти на альтернативные системы.

Можете ли Вы доверять пользователям в вопросе уважения Ваших интеллектуальных прав, пока Вы находитесь в бизнесе? Предоставляйте им необходимые услуги и возможности, и, надеюсь, они Вас не подведут. Если нет, то существует большая вероятность, что Вам ничего не удастся сделать, чтобы остановить их.

Выводы

Проблема состоит в том, что многие не понимают безопасность и того, насколько сложно обеспечить ее на хорошем уровне. К сожалению, существует огромное количество программ, которые поддерживают такое непонимание, реализуя неясности вместо настоящей безопасности. Возможно, Вы встречались с компаниями, которые обеспечивают услуги «восстановления данных», что означает *обход или слом предполагаемой системы безопасности неясных данных*.

Шифрование не является панацеей в системе безопасности. Если Вы не контролируете рабочую среду (оборудование, операционную систему и прочие работающие в системе программы), то у Вас нет контроля и над безопасностью - вне зависимости от схем шифрования, которые Вы используете. Это ситуация, когда Вы распространяете Вашу базу данных для установки на удаленных серверах.

Если Вам нужна настоящая защита данных или метаданных Вашей базы данных, то Вам необходимо сохранять контроль над файлом базы данных и над окружением, в котором происходит работа с этим файлом. Никакое другое решение не обеспечит такой же уровень безопасности.

Благодарности

Я хотел бы поблагодарить всех людей, которые просмотрели и прокомментировали эту статью. Также я хочу поблагодарить многих людей, которые осуществляют поддержку пользователй в листе поддержки Firebird: этот лист рассылки является источником бОльшей части информации, представленной в статье.

История документа

История переиздания

Н/д	14 фев 2005	GW	Первая редакция.
Н/д	11 апр 2005	GW	Пересмотрен раздел «Приемлемая низкая безопасность» с целью выделения простых алгоритмов XOR как слабых, чтобы убедить читателей изучать этот вопрос дальше, если есть интерес.
Н/д	26 апр 2005	GW	Дополнительный раздел по встроенному серверу (и ссылки на него). Сноска переведена в курсивное замечание, сноски не очень хорошо работают в HTML. Добавлено оглавление.
Н/д	4 дек 2005	GW	Добавлена ссылка на TrueCrypt. Добавлен раздел «Использование этого документа». Добавлен раздел благодарностей.
0.5	5 дек 2005	PV	Разделы «История документа» и «Использование этого документа» перемещены в приложения. Добавлен номер версии для использования в проекте Firebird. Документ добавлен в Firebird CVS.
0.5.ru	25 дек 2005	PM	Документ переведен на русский язык.

Использование этого документа

Я постарался сделать этот документ точным на момент написания, но я не могу гарантировать, что в нем нет ошибок. Безопасность является сложным предметом: если безопасность важна для Вашего проекта или инсталляции, Вам следует поискать совета профессионала.

Никаких особенных ограничений в использовании документа нет. Вы вольны воспроизводить, изменять или переводить этот документ. Однако измененные версии документа следует аннотировать относительно внесенных изменений и имени автора изменений, чтобы мое имя не ассоциировалось с текстом, который я не писал. - G.W.